On Routing Scalability in Flat SDN Architectures

Hemanth Kumar Ravuri*, Maria Torres Vega*, Jeroen van der Hooft*, Tim Wauters*, Bin Da†, Filip De Turck*
* Ghent University - imec, IDLab, Belgium
Email: hemanthkumar.ravuri@ugent.be
† Huawei Technologies Co., Ltd., China

***Abstract*—The rigidity of traditional network architectures, with tightly coupled control and data planes, impair their ability to adapt to the dynamic requirements of future application domains, such as the Tactile Internet or Holographic-Type Communications. Software-Defined Networking (SDN) architectures, which provide programmability to configure the network, have the potential to provide the required dynamism. However, given its centralized essence, SDN suffers from scalability issues. Therefore, efforts have been made to propose alternative decentralized solutions, such as the flat distributed SDN architecture. Despite its potential, the real applicability and scalability of decentralized SDN solutions are still open research questions. This paper presents a comparative analysis of the effects of different routing approaches on the scalability of flat distributed SDN architectures. Using the Open Network Operating System (ONOS) as our evaluation architecture, we have studied the tradeoff between routing overhead in the control data plane and inter-controller communications for different degrees of decentralization. We have found that routing applications, which only require control-data plane communication for setting the path, benefit more from decentralization than the ones which utilize inter-controller communications and ensure Quality of Service (QoS). Our findings highlight the need for efficient routing mechanisms to deal with inter-controller overhead while lowering the amount of control-data plane communication.**

***Index Terms*—ONOS, controller, distributed, QoS, overhead**

I. Introduction

In the era of high-speed communications, futuristic application domains enforce stringent requirements on the network infrastructures. Applications such as immersive multimedia delivery, Holographic-Type Communications (HTC) or the Tactile Internet envision ultra-high bandwidth (in the range of Gbps) and ultra-low latency (in the order of milliseconds) [1]. Traditional network architectures, with tightly coupled control and data planes, are hindered by their inability to dynamically adapt and enforce control policies based on the requirements. Software-Defined Networking (SDN) architectures, on the other hand, have the potential to provide the required dynamism. SDN decouples the control and data planes to achieve a logically centralized control architecture. Therefore, it enables programmability to configure the networks according to the dynamic needs of the applications [2]. It further allows the operators to develop and implement application-specific functionality, such as routing strategies.

Current SDN architectures are mostly centralized. However, the centralization can lead to scalability issues [2]. For instance, if the load of the controller reaches a certain threshold, the request-processing latency will substantially increase. This becomes even more challenging if the controller runs computational-intensive mechanisms involving multipath routing algorithms. Thus, efforts have been made to propose alternative distributed control plane architectures [2]. One clear example is the flat distributed architecture, where the network is horizontally partitioned into multiple domains, each of them handled by a controller. Such architecture helps in the sharing of load and improves resiliency.

The application and performance of such architectures in terms of scalability have not been experimentally evaluated. The purpose of this paper is to provide a comparative analysis of the effects of different routing approaches on the scalability of flat distributed SDN architectures. Towards this objective, we have selected three different representative routing approaches based on the communication overhead for setting up the flow. These are the Reactive Forwarding application (Fwd), the Intent-Based Reactive Forwarding Application (iFwd) [3] and the Multipath Routing Application (MRA). To perform the evaluation, we have chosen the Open Network Operating System (ONOS) controller [4].

Our findings show that routing approaches where overhead is mainly in the control-data plane communication (Fwd) benefits more from decentralization than inter-controller communication-based solutions (iFwd and MRA). However, the hop-by-hop mechanism of Fwd proves to be detrimental to scalability in terms of gains obtained by deploying additional controllers. Nevertheless, iFwd provides a means to deal with overhead between the control-data plane by providing a consensus between controllers. There is a need for an efficient routing mechanism to deal with the inter-controller overhead to support applications like MRA in meeting the QoS requirements.

In the remainder of this paper, we briefly describe the state-of-the-art on analysis of routing protocols in decentralized networks. Subsequently, the proposed methodology and results are presented and conclusions are drawn.

II. Background

As introduced previously, to mitigate the issue of SDN controller scalability, the present research tendency is toward decentralizing the control plane. Prominent among such approaches is the flat distributed architecture. ONOS is a representative of such an architecture [4] aimed at providing high performance, availability and scalability. The key component of ONOS is the distributed core, which enables the consistency among multiple controller instances in a cluster. The controllers in a cluster are always logically connected in a

full mesh, using a specific TCP port (9876). The ONOS core provides various services through its subsystems [4], such as the device, the link, the host, and the topology subsystems. One of the most important aspects of ONOS is its distributed storage. It enables each controller in the cluster to update the state information on the distributed data stores. Further, its architecture supports the intents framework which allows the applications to submit their requirements in the form of policy-based desires known as intents. Further, ONOS outperformed competitors like OpenDaylight during various benchmark evaluations [5] making it the best-suited candidate of flat SDN architecture for our routing scalability evaluation.

Various works have focused on benchmarking the controller scalability using tools such as 'Cbench' and 'HCProbe' [5], [6], considering several well-known SDN controllers. Such tools only benchmark the threading scalability of the in a single controller scenario. Few works focused on proposing models to estimate different forms of overhead in the ONOS architecture. Bianco *et al.* propose a quantitative model for ONOS cluster to estimate the number of messages exchanged between the control plane and data plane [3]. However, to the best of our knowledge, none of the previous works evaluated the impact of such overhead on the routing scalability of the architecture. ONOS releases a set of results, benchmarking its sub-components using Cbench. However, they only look at the throughput of each subsystem. Our work measures the scalability for routing mechanisms, which involves multiple subsystems working together. A work evaluating the scalability of flat distributed controllers for various routing scenarios is still missing in the current literature. Our work aims at filling that void, by evaluating its scalability using three different routing applications on two real-world topologies under varied load conditions.

III. Routing Mechanisms in ONOS

This section presents the key routing mechanisms utilized in this study of scalability of flat distributed SDN architectures. In SDN, a routing application aids the controller to reactively [3] handle the incoming flow requests based on the defined set of policies. When selecting the routing applications that we would put under test, we aimed to understand where a mechanism could become a bottleneck when exchanging messages. The three routing mechanisms have been used and are presented as follows (Figure 1).

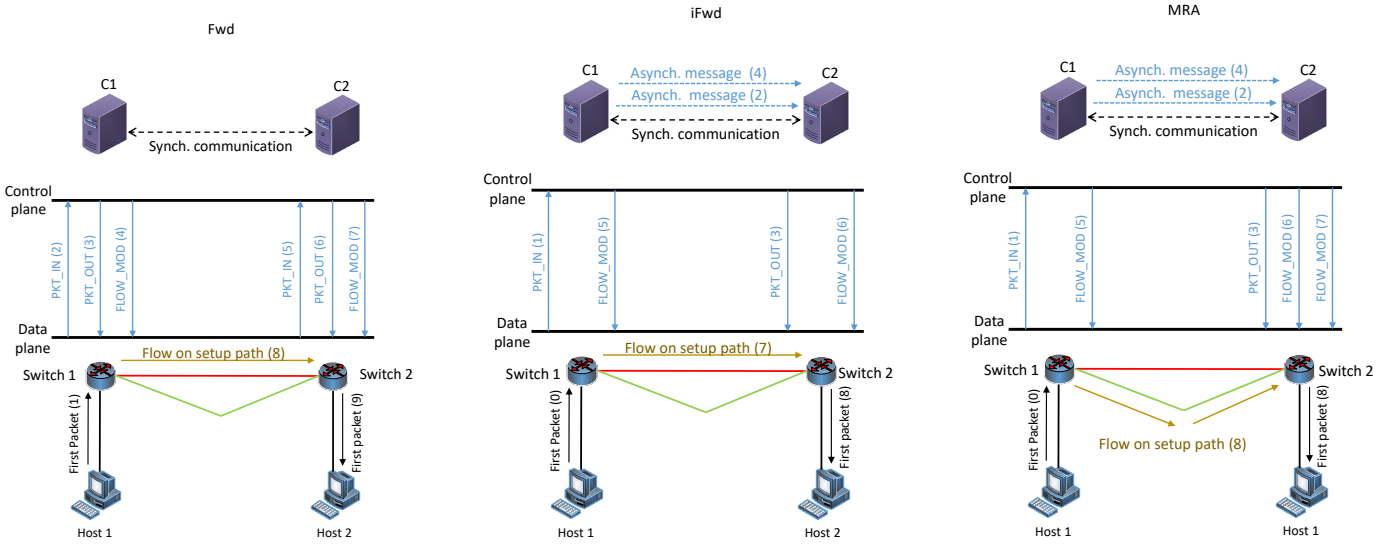
1) Reactive Forwarding Application (Fwd) - High Control-Data Plane Communication Overhead: Figure 1a illustrates the flow setup process when Fwd is used for two switches where each switch is handled by a separate controller. The complete flow setup process is as follows [6]. When host 1 tries to communicate with host 2, the first packet of the new flow arrives at switch 1, which searches in its flow table for matching flow rules. If no prior flow rows are installed, it sends a PKT_IN message to controller 1 (C1). Assuming that C1 is aware of the position of both source and destination hosts, it requests the path service to provide the shortest path between them. Subsequently, C1 sends the packet back to switch 1 with a PKT_OUT message, instructing it to forward the packet to the switch 2 through a specific port. Further, C1 sends a FLOW_MOD message to switch 1 to install a flow rule to enable the routing of all the following packets in the flow. After installing the flow rule, Switch 1 forwards the packet to switch 2. Subsequently, Switch 2 sends a new PKT_IN message to controller 2 (C2). C2 sends a PKT_OUT message to switch 2 instructing it to send the packet to host 2 and then sends a FLOW_MOD to forward subsequent packets. The hop-by-hop process is repeated for all the switches on the path.

2) Intent-Based Reactive Forwarding Application (iFwd) - High Inter-Controller Communication Overhead: The iFwd application is based on the intent mechanism in ONOS, which makes use of inter-controller communication to set up paths. Figure 1b illustrates the flow setup process [3]. Switch 1 sends a PKT_IN message to C1 when there is no matching flow rule for the first packet of the flow which requests the intent service to provide a host-to-host intent. The intent service selects a path and computes an intent. Since there is more than one controller, C1 requests the mastership service to provide information on C2, responsible for the destination switch. Then C1 sends an asynchronous message to C2 (through port 9876) with the context of the initial packet and information about the destination. After receiving the packet, C2 sends a PKT_OUT message to switch 2 instructing it to forward the packet to the destination host. C1 further shares the intent information with C2. Finally, intents are translated into flow rules and installed on the switches using FLOW_MOD messages.

3) Multipath Routing Application (MRA) - Ensuring QoS: The MRA is based on the Widest Shortest Path algorithm [7]. This application chooses the path with the highest amount of bandwidth available, *i.e.*, the widest path. MRA makes use of the topology, device and link services to keep track of the link bandwidth utilization based on a defined monitoring period. Such an approach provides dynamism to the application as it makes use of traffic information. As seen in Figure 1c, the flow setup process is very similar to that of iFwd except that there is an additional step for identifying the widest shortest path. When there is congestion (red line) on the shortest path, MRA sends the flow along the path with higher bandwidth (green), while Fwd and iFwd send it on the congested one (Figure 1). It should be noted that in all three cases, the controllers exchange synchronous messages periodically (five seconds by default) to maintain a consistent global view using the anti-entropy protocol. In addition, in case of Fwd each switch on the path needs one PKT_IN and one PKT_OUT message, the total control-data plane overhead to set up a path of length x is $2x$. In case of iFwd and MRA, this overhead equals only 2. Flow_Mod messages are common in all the flows, so we are not considering it as part of the overhead.

IV. Experimental Evaluation

This section presents our experimental setup, the results, followed by the insights and conclusions.

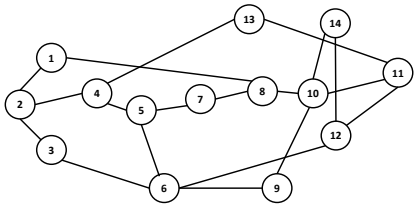


(a) Fwd routing application flow chart. (b) iFwd routing application flow chart. (c) MRA routing application flow chart.

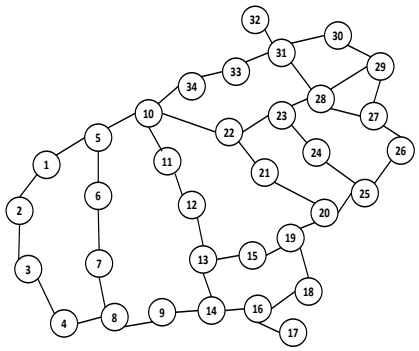
Fig. 1: Flow charts describing the routing applications to be tested.

Topo	N.Con	N.Sw	N.H
NSFnet	1	C1: 1-14	1 to 10
NSFnet	2	C1: 1-7 C2: 8-14	1 to 10
NSFnet	3	C1: 1-4 C2: 5-9 C3: 10-14	1 to 10
Internet2	1	C1: 1-34	1 to 5
Internet2	2	C1: 1-17 C2: 18-34	1 to 5
Internet2	3	C1: 1-11 C2: 12-23 C3: 24-34	1 to 5

(a) Table



(b) NSFNET topology



(c) Internet2 OS3E topology.

Fig. 2: Controller-switch assignments and number of hosts per switch, for both the topologies in different controller cases.

A. Experimental Setup

In order to evaluate the scalability of the three routing protocols on ONOS, we have emulated the network using Mininet[1] for two different topologies: NSFnet with 14 switches [7], and Internet2 OS3E with 34 switches [2] (Figure 2). Distribution has been evaluated with three controller scenarios, namely one controller (centralized), two and three controllers. Switches are assigned to each of these controllers based on the ONOS default load balancing application [4]. Further information on controller-switch assignments per topology and the number of hosts per switch can be seen in Table 2a. In order to provide an analysis of the effects of load on the scalability of the architecture and routing application, the load has been varied by increasing the number of hosts per switch (from one to ten for the NSFnet topology and from one to five for Internet2 OS3E). The load has been emulated by generating ping flows between all the pairs of hosts which increases the number of generated flow requests quadratically. We have used the tool Fping[3] to generate these flows. To further vary the load, the inter-arrival time between the flows has been set to 1 or 5 ms. The monitoring period of the controller has been fixed to ten seconds. We have implemented ONOS controllers on docker containers. We have carried out the experiments with the Mininet emulation and controllers running on two separate nodes, each with a quad-core Intel(R) CPU E5520 @ 2.20GHz and 12 GB of RAM. Scalability has been measured as the aggregated value of the percentage of accepted flows. Each experiment was repeated ten times and the values were aggregated.

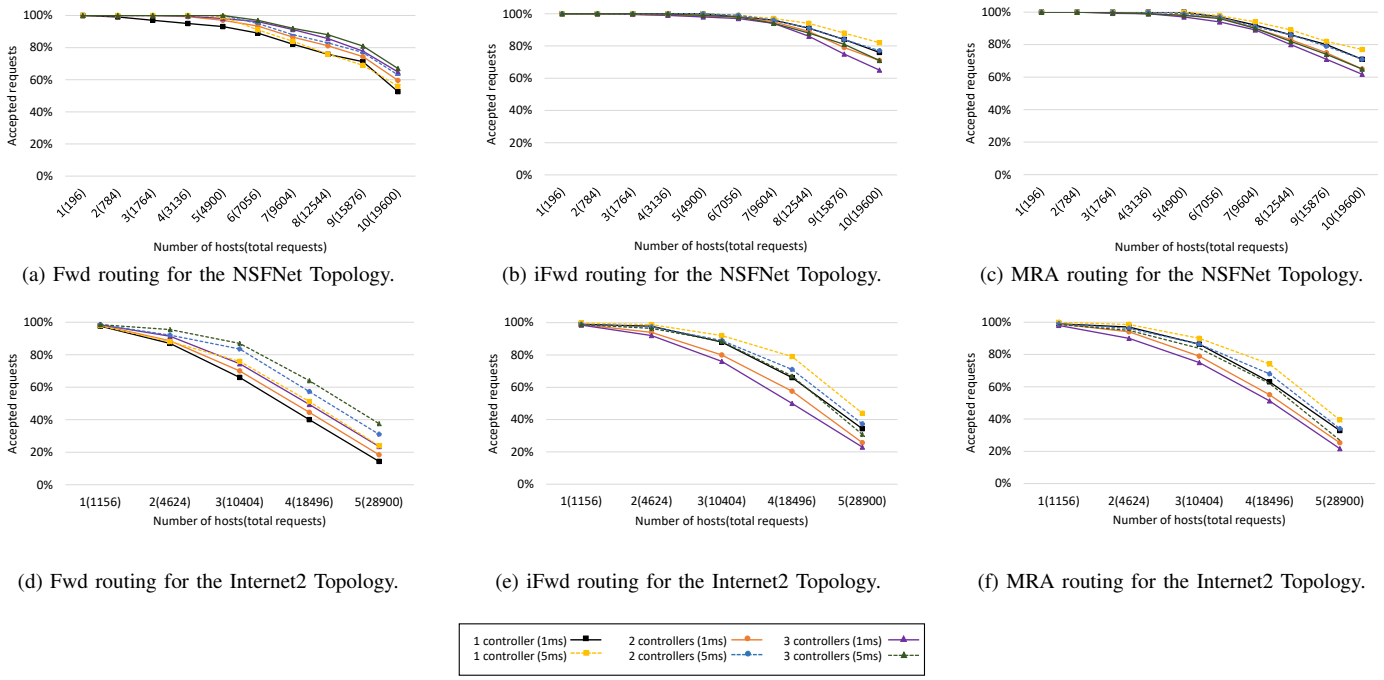
B. Results

As the first step of this analysis, we aimed to evaluate the scalability of the three routing protocols for two network

[1]http://mininet.org/

[2]https://noc.net.internet2.edu/

[3]https://fping.org/fping.1.html



(a) Fwd routing for the NSFNet Topology.

(b) iFwd routing for the NSFNet Topology.

(c) MRA routing for the NSFNet Topology.

(d) Fwd routing for the Internet2 Topology.

(e) iFwd routing for the Internet2 Topology.

(f) MRA routing for the Internet2 Topology.

(g)

Fig. 3: Scalability of routing techniques for different controller cases in two topologies, for an inter-arrival time of 1ms and 5 ms under varying hosts per switch (total number of requests).

topologies for different load conditions. Figure 3 shows the results of this evaluation. First of all, the scalability (*i.e.*, the percentage of accepted requests) decreases with the increase in the number of requests sent (presented next to the number of hosts per switch) independently from the routing application and the number of deployed controllers. In addition, for each case, the percentage of accepted requests decreases as the inter-arrival time is reduced from 5 ms to 1 ms. This shows the impact of the load on the controller scalability. One clear example is the case of the Fwd application applied to the Internet2 topology with five hosts per switch. When the inter-flow arrival time is reduced to 1 ms, the percentage of requests accepted drops from 23.9% to 14.3% in the case of one controller (centralized) and from 37.6% to 23.5% for the three controllers case. When the Fwd application is deployed (Figures 3a and 3b), the scalability performance improves with the number of controllers. For example, decentralizing the network up to three controllers brings an improvement of 13.7% for the Internet2 OS3E case with five hosts per switch and inter-flow arrival time of 5 ms. The performance of iFwd (Figures 3c and 3d) shows an intrinsically different behavioral pattern. The scalability is observed to degrade with the number of controllers. Taking the same configuration as before, increasing the number of controllers from one to three reduces scalability by 12.9%. Finally, Figures 3e and 3f present the results for the MRA. Since the flow setup process is very similar to that of iFwd, the results show a comparable trend.

The improvement in scalability with decentralization in the case of Fwd is due to the sharing of the control plane-data plane overhead among the controllers. As discussed in Section III, the overhead generated by Fwd between the control plane and data plane increases with the number of flows and length of the path. However, the improvement is not very high due to the hop-by-hop approach of Fwd. Handling one PKT_IN message does not guarantee an end-to-end flow as each switch on the path generates its own PKT_IN message but dropping one PKT_IN message can make the flow incomplete. Given the variable length of the paths, the huge number of generated PKT_IN messages leads to incomplete flows, thus not completely converting into a substantial number of accepted flows. In case of iFwd and MRA, the opposite has been observed with decentralization because of the communication overhead between the controllers for every flow request. As discussed in Section III each flow setup involves sharing of the packet and intent information among the controllers along the path. With an increase in the number of controllers, the network partition becomes smaller, leading to increased inter-controller communication due to the higher number of inter-partition flows. The single controller case does not involve such an extra overhead.

In order to further explore the tradeoffs between the overhead created by iFwd and MRA, and the scalability improvements of Fwd, we fix the load at the highest value (ten hosts per switch in NSFnet topology, five hosts per switch in Internet2 OS3E topology and inter-flow arrival time of 1 ms). The performance was again evaluated in terms of scalability (percentage of accepted flow requests). As the number of controllers grows, the scalability increases by close to 10.0% in both topologies for Fwd (Figure 4). In the case of iFwd, the scalability falls by 11.0% in both the topologies.

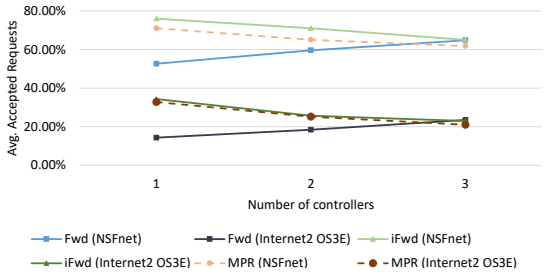


Fig. 4: Reactive forwarding versus Intent-based reactive forwarding vs Multipath routing.

As discussed in Section III, the overhead generated by Fwd between the control plane and data plane (in terms of the number of PKT_IN and PKT_OUT messages) is twice the length of the path, while in the case of iFwd, we only have one pair of messages irrespective of the length of the path. Even though the overhead between the control plane and the data plane is lower in the case of iFwd, the presence of additional inter-controller overhead per-flow impacts its scalability. The average throughput between two controllers during the experiment is observed to be 32 kbps for Fwd (needed for synchronization periodically) and 278 kbps for iFwd (for asynchronous communication). The inter-controller overhead for iFwd is almost nine times that of Fwd for just one controller pair. The exchange of messages during the flow setup process also incurs delay. This overhead has become three times when all pairs of controllers involved in the flow setup are considered. Hence, the performance depreciates for iFwd with an increase in the number of controllers. As introduced in Section III and hinted at while analyzing Figure 3, MRA has the advantage of choosing the best path in terms of available bandwidth, while following the flow setup mechanism of the iFwd solution. However, iFwd performed slightly better than MRA, irrespective of the number of controllers in both topologies (up to 5%). This is because MRA involves an additional process of computing the widest shortest path per request.

From the results we infer that one single controller can be a performance bottleneck (Figures 3a and 3b). Therefore, moving towards a flat distributed SDN control plane improves scalability. This was observed in the Fwd case. However, the hop-by-hop approach is not desirable. An alternative reactive mechanism with reduced overhead is needed. The performance of iFwd degrades with the number of controllers due to the inter-controller overhead. So in a flat distributed SDN architecture, the presence of such overhead for every flow is highly undesirable. The scalability further decreased for the MRA, due to additional processing overhead. While lower control plane and data plane overhead is desirable, the observed results call for an efficient mechanism to deal with inter-controller overhead when setting up paths. While iFwd provides the former, a technique that reduces the inter-controller overhead per flow is needed. One way forward is to set up flows along the shortest path within each controller's network partition and use an eventually consistent model to reroute a batch of flows on the best path. Such a model, with the support of adaptive mechanisms taking into account dynamic traffic conditions, will reduce the inter-controller overhead per-flow setup while optimizing the number of flows handled during the synchronization cycle.

V. Conclusions and Future Work

In this paper, we have evaluated the scalability of three different routing applications in flat Software-Defined Networking (SDN) control plane architectures, using ONOS as the representative controller. We have considered two real-world topologies and compared the centralized to distributed cases. Our results show that the scalability improves with the number of controllers when the routing techniques involve overhead between the control plane and the data plane. However, it deteriorates for routing techniques with higher inter-controller overhead and Quality of Service (QoS) control. Nevertheless, iFwd provides a means to reduce the overhead between the control and the data planes. An intelligent mechanism to reduce the inter-controller overhead per flow is needed. Such a mechanism can help dealing with flows having specific QoS demands. Therefore, this study calls for an efficient mechanism to deal with inter-controller overhead while setting up paths, without increasing the control-data plane communication. Further, as our future work, we will focus on evaluating the source routing techniques like the segment routing [8] for flat distributed SDN controller architectures. In such techniques the path information is embedded inside the packet header itself. We will further evaluate the scalability of the hierarchical SDN control plane architecture for different routing scenarios.

Acknowledgment

This research is the result of a collaborative project between Huawei and Ghent University, and funded by Huawei Technologies, China. Maria Torres Vega is funded by the Research Foundation Flanders (FWO), grant number 12W4819N.

References

[1] A. Clemm, M. Torres Vega, H. K. Ravuri, T. Wauters, and F. De Turck, "Toward truly immersive holographic-type communication: Challenges and solutions," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 93–99, 2020.

[2] F. Bannour, S. Souihi, and A. Mellouk, "Distributed sdn control: Survey, taxonomy, and challenges," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 333–354, 2018.

[3] A. Bianco, P. Giaccone, R. Mashayekhi, M. Ullio, and V. Vercellone, "Scalability of onos reactive forwarding applications in isp networks," *Computer Communications*, vol. 102, pp. 130–138, 2017.

[4] "System components - onos - wiki." [Online]. Available: https://wiki.onosproject.org/display/ONOS/System+Components (accessed Apr. 29, 2020).

[5] L. Zhu, M. M. Karim, K. Sharif, F. Li, X. Du, and M. Guizani, "SDN controllers: Benchmarking & performance evaluation," *CoRR*, vol. abs/1902.04491, 2019.

[6] H. K. Ravuri, M. Torres Vega, T. Wauters, B. Da, A. Clemm, and F. De Turck, "An experimental evaluation of flow setup latency in distributed software defined networks," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 432–437.

[7] S. Tomovic and I. Radusinovic, "Fast and efficient bandwidth-delay constrained routing algorithm for sdn networks," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, 2016, pp. 303–311.

[8] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The segment routing architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.